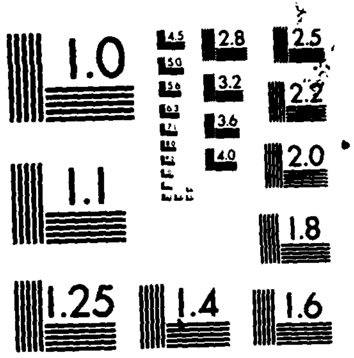
AD-A172 041 SIMULATION OF AN ULTRACOMPUTER WITH SEVERAL 'HOT SPOTS' 1/1
(U) STANFORD UNIV CA DEPT OF COMPUTER SCIENCE
D S ROSENBLUM ET AL JUN 86 STAN-CS-86-1119
UNCLASSIFIED N00014-85-C-0731 F/G 9/2 NL


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

June 1986

Report No. STAN-CS-86-1119 (12)

Simulation of an Ultracomputer with Several 'Hot Spots'

by

David S. Rosenblum and Ernst W. Mayr

Department of Computer Science

Stanford University
Stanford, CA 94305

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A172 041	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Simulation of an Ultracomputer with Several 'Hot Spots'		5. TYPE OF REPORT & PERIOD COVERED technical
		6. PERFORMING ORG. REPORT NUMBER STAN-CS-86-1119
7. AUTHOR(s) David S. Rosenblum, Ernst W. Mayr		8. CONTRACT OR GRANT NUMBER(s) N00014-85-C-0731
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Department, Stanford University, Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research (Code 458), Arlington, VA 22217		12. REPORT DATE June 1986
		13. NUMBER OF PAGES 34
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) ONR Representative - Mr. Robin Simpson, Durand Aeronautics Building, Rm. 165, Stanford University, Stanford, CA 94305		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release: distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Parallel Computation, Memory Access, Synchronization, Ultracomputer		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) See next page		

DD FORM 1 JAN 73 1473

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19. KEY WORDS (Continued)

20 ABSTRACT (Continued)

This report describes the design and results of a time-driven simulation of an Ultracomputer-like multiprocessor in the presence of several "hot spots," or memory modules which are frequent targets of requests. Such hot spots exist during execution of parallel programs in which the several threads of control synchronize through manipulation of a small number of shared variables. The simulated system is comprised of N processing elements (PEs) and N shared memory modules connected by an $N \times N$ buffered, packet-switched Omega network.

The simulator was designed to accept a wide variety of system configurations to enable observation of many different characteristics of the system behavior. We present the results of four experiments: (1) General simulation of several 16-PE configurations, (2) General simulation of several 512-PE configurations, (3) Determination of critical queue lengths as a function of request rate (512 PEs) and (4) Determination of the effect of hot spot spacing on system performance (512 PEs).

Simulation Of An Ultracomputer With Several 'Hot Spots'

David S. Rosenblum
Computer Systems Laboratory
Department of Electrical Engineering

Ernst W. Mayr
Department of Computer Science
Stanford University

Abstract

This report describes the design and results of a time-driven simulation of an Ultracomputer-like multiprocessor in the presence of several "hot spots," or memory modules which are frequent targets of requests. Such hot spots exist during execution of parallel programs in which the several threads of control synchronize through manipulation of a small number of shared variables. The simulated system is comprised of N processing elements (PEs) and N shared memory modules connected by an $N \times N$ buffered, packet-switched Omega network.

The simulator was designed to accept a wide variety of system configurations to enable observation of many different characteristics of the system behavior. We present the results of four experiments: (1) General simulation of several 16-PE configurations, (2) General simulation of several 512-PE configurations, (3) Determination of critical queue lengths as a function of request rate (512 PEs) and (4) Determination of the effect of hot spot spacing on system performance (512 PEs).

This work was supported in part by a grant from AT&T, ONR contract N00014-85-C-0731, NSF grant DCR-8351757, DARPA contract N00039-84-C-0211 and NASA grant NAGW-419.

1 Introduction

This report describes the design and results of a time-driven software simulation of an Ultracomputer-like multiprocessor comprised of N processing elements (PEs) and N shared memory modules connected by an $N \times N$ packet-switched Omega network; a familiarity with the basic design of the NYU Ultracomputer and the Omega network as described by Gottlieb *et al.* [GGK83] is assumed throughout. The simulated system incorporates one minor restriction of the Ultracomputer model, namely that request packets are allowed to participate in only one combining operation at a network switch; of course, packets may participate in several combining operations as long as they occur at *different* network stages.

The simulator is an Ada[1] [DoD83] program of around 1500 lines subdivided into three packages and a main driver procedure. The program was compiled, debugged and executed using a validated Ada compiler running on the Data General MV10000 operated by the Program Analysis and Verification Group of the Computer Systems Laboratory at Stanford University.

The simulator was specially designed to reveal certain aspects of the behavior of the Ultracomputer in the presence of a small number of "hot spots" or shared memory modules which are frequent targets of data requests. Such hot spots would exist during execution of parallel programs in which constituent threads of control synchronize through modification of a small set of globally shared variables (such as semaphores or condition variables). The statistics of interest output by the simulator include average and maximum request latency, average and maximum queue lengths in each Omega network stage, and actual numbers of queues and wait buffers used in each Omega network stage; these statistics are computed empirically as functions of input parameters to the simulation, such as maximum allowed queue lengths, memory request rate and number of hot spots present.

The major features which set our work apart from previous efforts include (1) Packet combination, (2) Wait/no-wait policies for packet creation (described below), (3) Multiple numbers of hot spots, (4) Variability of hot spot assignment and spacing and (5) Fair and robust solution of conflicts. These features and all results will be described more fully in later sections.

Kruskal and Snir [KS83] have derived some asymptotic results, both analytically and through simulation, for bandwidth in both unbuffered and buffered systems connected by various Banyan networks; an "unbuffered" system can be considered to be equivalent to a buffered system with queue lengths of one. Their simulations do not incorporate packet combining, but instead rely on packet dele-

[1]Ada is a registered trademark of the US Government-AJPO.

tion to solve conflicts. The simulations described in this report allow combining of packets, and, as described below, packets are never deleted once they have been enqueued at a PE. In particular, packet movement is held up if necessary to await available queue space, and packets are deleted only upon creation if either the PE queue is full or if a wait policy is being used. One feature of the simulations of Kruskal and Snir *not* allowed for in our effort is the simulation of time-multiplexed, multi-packet requests.

Pfister and Norton [PN85] performed simulations of systems connected by buffered Omega networks. They simulated the presence of variable traffic to a single hot spot superimposed over a background of light, uniform traffic. The motivation for their work was to determine whether or not the cost of added packet combining hardware is justified by a relative increase in performance; they concluded that the extra cost can be justified, especially because combining reduces the phenomenon of network congestion which they term "tree saturation." They claim that wait buffers of length six and queues of length four give "adequate" performance, apparently for all systems and applications. Furthermore, they claim that "real code" typically develops only one or two hot spots, and that results for systems with multiple hot spots are "essentially identical" to their results for systems with a single hot spot. Our simulations demonstrate that a *single* queue size is *not* adequate for all offered loads (i.e., request rates), but instead is either wasteful or insufficient for those loads not within the small range for which the single size is ideal; however, our simulations do demonstrate that for a fixed load, system performance is not very dependent on the number of hot spots present or their relative placement.

The first section below gives a detailed description of the algorithms executed by the simulator along with a description of the input and output operations performed. This is followed by a brief examination of possible alternatives to some of the chosen design decisions which have an effect on the output statistics. Finally, results are presented for the simulation of both a 16-PE Ultracomputer and a 512-PE Ultracomputer.

2 The Simulation Algorithm

In order to make the simulator as flexible as possible in producing desired statistics, each simulation run is parameterized by the following input data values requested from the terminal:

- Number of simulation steps to perform.
- Seed for the pseudo-random number generator.

- Number of PEs/memory modules.
- Number of hot spots.
- Maximum number of hot spots to assign to each PE.
- Whether or not to choose hot spot locations randomly.
- Spacing, in numbers of memory modules, of hot spots if not chosen randomly.
- Maximum allowed deviation from the spacing value for hot spots if not chosen randomly.
- Whether or not to allow multiple outstanding requests to the same hot spot.
- Probability of assigning a hot spot to a PE.
- Maximum length of all queues.
- Probability of a memory request.
- Filename for statistics output.

The program can be easily extended to accept three other input parameters:

- Switchbox size (currently assumed to be 2×2)
- Probability of generating a new hot spot (currently assumed to be 0.0 and, thus, ignored)
- Probability of removing an existing hot spot (currently assumed to be 0.0 and, thus, ignored)

For the purposes of the simulation, we assume that all memory requests are Fetch&Adds, and that each memory module contains a single memory word, the location of the hot spot.

The simulation consists of four phases:

1. Initialization of the virtual Ultracomputer.
2. Main simulation steps, during which memory requests are generated and fulfilled.
3. Final simulation steps, during which all remaining outstanding requests requests are fulfilled.

4. Output of statistics.

Each of these phases is examined in detail below. The simulator uses a generator of uniformly pseudo-random positive integers to determine when to perform certain actions which are guided by the input probabilities. For the remainder of this report, let request packets be called MM packets when travelling from PEs to memory modules and PE packets when travelling from memory modules to PEs. Also, let the Omega network stage closest to the PEs be numbered 0; the stage closest to the memory modules is then numbered $K-1$, where $K = \log_2 N$ (since 2×2 switching elements are assumed).

2.1 Initialization

During the initialization phase, the simulator first requests all of the input parameters from the terminal, reprompting the user when illegal values are encountered. Next, the simulator seeds the pseudo-random number generator using the value input from the terminal. Then, the hot spot addresses are chosen.

Let H be the total number of hot spots to be simulated. If the hot spot addresses are to be chosen randomly, the pseudo-random number generator is used to randomly choose addresses until exactly H *different* hot spots have been chosen. Otherwise, hot spot addresses are chosen according to input spacing parameters S (the spacing value) and V (the maximum magnitude of deviation from S); the input routines check these values to insure that $H * (S + V) \leq N$, where N is the number of memory modules. The first hot spot address is chosen at random; the remaining $H - 1$ hot spot addresses are chosen iteratively as follows, wrapping around to the beginning of the address space if necessary: The next address is chosen by adding an increment to the previously-chosen address; the initial value of the increment is S, to which is added a value chosen randomly from the interval $[-V, V]$ to obtain the final increment. This iteration is performed until exactly H different hot spots have been chosen.

Finally, the hot spot assignment table is set up as follows using the hot spot density parameter D (maximum hot spots per PE) and hot spot assignment probability P as input to the simulator: D assignment rounds are performed, and within each round each PE is assigned, with probability P, a randomly chosen hot spot (from the set of chosen hot spot addresses) different from all hot spots previously assigned to the PE. Thus, the average number of hot spots assigned to each PE is DP.

2.2 Main Simulation Steps

All simulation runs are clock-driven; each simulation step (equivalent to one tick of the simulator clock) corresponds to a single Ultracomputer cycle and consists of three main substeps: (1) At most one new memory request is generated and enqueued for each PE according to the generation probability input to the simulator; (2) All other outstanding requests are moved forward at most one Omega network stage in their paths according to the Ultracomputer algorithm [GGK83], taking into account full queues which may delay routing; (3) Queue length statistics (maxima and cumulative sums) are updated.

Requests are generated (and appropriate sums updated for statistics) during a simulation run according to one of two policies (determined by a simulator input parameter): a "no-wait" policy in which multiple outstanding requests to the same hot spot by a single PE are allowed, and a "wait" policy in which a PE must wait for requests to a hot spot to be satisfied prior to generation of a new request to the same hot spot. If Z is the number of simulation steps and G is the probability of a PE generating a request at each step, then the mean number of requests generated during a simulation run using a "no-wait" policy is ZGN. Each request generated for a PE is destined for a randomly chosen hot spot, but if the "wait" policy is in effect, the request is discarded if the currently PE has an outstanding request to the chosen hot spot.

The simulated movement of request packets is carried out in what will be referred to in the rest of this paper as a "forward simulation." That is, MM packets are moved in a stage by stage manner, first from the PEs to Omega network stage 0, then from stage 0 to stage 1, then from stage 1 to stage 2, and so forth until MM packets are moved from stage $K-1$ to the memory modules, where the Fetch&Adds are performed and the return PE packets are enqueued; next PE packets are routed from the memory modules to stage $K-1$, then from stage $K-1$ to stage $K-2$, and so on until PE packets are moved from stage 0 to the PEs and then destroyed after updating some statistics. A "backward simulation" would visit the stages in the reverse order, first moving PE packets from stage 0 to the PEs, then moving PE packets from stage 1 to stage 0, etc., until finally newly generated MM packets are moved from the PEs to stage 0. To ensure that routing preference is not given to packets moving through smaller numbered ports in a network stage, routing is performed on a switchbox-by-switchbox basis in the destination stage of each step. This "fair" routing scheme is described further below.

Movement of packets between stages occurs only if queue space is available at the target stage. To ensure that a packet is routed at most one stage per simulation step, each packet is timestamped with the value of the simulator clock each time it is moved; these timestamps are then compared to the simulator clock value and used in the obvious way. In addition, requests may be combined at a switching element according to the Fetch&Add algorithm described by Gottlieb *et al.* [GGK83]; one wait buffer pool is associated with each MM output port of each switching element. The MM packets resulting from a combine operation may participate in further combine operations in later network stages only, but packets may participate in at most one combine operation at a single stage. Thus, there will be at most one wait buffer entry per stage for each request.

To make the routing fair, switchboxes (or memory modules or PEs) in the *destination* stage of a routing substep are visited sequentially. All queues and wait buffer pools are either infinite or are of the same finite length if a maximum length is input to the simulator. Queue space is allocated as follows to each pair of MM packets entering a switching element: if both incoming packets are destined for the same output port of the switchbox, then they are combined if port queue space and wait buffer space is available in the destination switchbox, or else one is selected randomly for routing if only port queue space is available; otherwise (different output ports) each one is routed only if port queue space is available. Queue space is allocated to each pair of PE packets entering a switching element by taking each packet in a random order and performing the routing only if queue space exists in the destination switchbox for the packet and its corresponding wait buffer entry (if one exists). In all cases, once a packet is moved, combined or "uncombined," the old buffer space becomes available for immediate use by other packets being routed.

2.3 Final Simulation Steps

The final simulation steps are performed exactly as in the previous section except that no new memory requests are generated; simulation steps are performed only until all outstanding requests have been satisfied.

2.4 Statistics Output

Statistics are computed and output to the file specified in the input to the simulator. Averages are computed by dividing cumulated sums by the requested number of simulation steps, ***not*** by the total number of simulation steps performed; i.e., finalization steps are not counted when computing averages. The following statistics are computed and output:

- Starting and ending time-of-day for the simulation.

- Number of extra (finalization) simulation steps performed.
- Memory module numbers chosen as hot spots.
- Average number of hot spots assigned to each PE.
- Total number of requests generated.
- Average number of requests per PE.
- Average, minimum and maximum numbers of simulation steps needed to complete a request.
- Average length of the queues at the PEs (holding newly-generated MM packets) at the end of a simulation step.
- Maximum length of the queues at the PEs at any time.
- Average and maximum lengths of the queues at the hot spots (holding returning PE packets).
- Numbers of MM port queues, PE port queues and wait buffer pools used in the Omega network, calculated for each network stage.
- Average and maximum lengths of MM port queues, PE port queues and wait buffers in the Omega network, calculated for each network stage.

3 Design Tradeoffs Affecting Results

This section briefly analyzes a few of the simulator design choices and the differing effects these choices and their alternatives may have had on the generated statistics.

3.1 Forward vs. Backward Simulation

In designing the main simulation step, it was observed that the execution of a backward simulation would simulate an Ultracomputer which allowed all packets to be routed synchronously and simultaneously one stage forward in their paths. For example, queue space in stage I freed as a result of routing MM packets to stage $I+1$ could be used immediately within the same routing step for packets coming from stage $I-1$; applying this reasoning globally, the net result is that the simulated system would be able to propagate queue availability information in zero time all the way from the end of the request packet path (PE packet output port queues in stage 0) back to the starting point of the request packet path (MM packet queues within the PEs).

Because this assumption of zero propagation delay is completely unrealistic for large Ultracomputer configurations built using current digital device technology, the forward simulation method was chosen, thus giving the most conservative picture of the Ultracomputer behavior. This choice of method affects the simulation results most notably in simulations of heavy loads using finite queues; using the forward method, the average packet can require as much as twice as many steps as would be needed in the backward method, since queue space cleared in one step cannot be used until the next step. That is, using the forward method on a loaded system of finite queues, a packet moves forward one queue entry per two simulation steps in the worst case.

3.2 Wait vs. No-Wait Request Generation

It was decided to allow simulation of both a wait and a no-wait request generation policy because of the different PE characteristics these two policies imply. A wait policy simulates approximately a parallel program execution environment on PEs with relatively long times between process switches, few constituent threads of control from each executing program and/or no sharing of hot spots among processes. A no-wait policy simulates approximately a set of PEs with short times between process switches, multiple constituent threads of control from each executing program and/or hot spots shared among processes.

The effect of the choice of generation policy used in a simulation run is most noticeable when a wait policy is used and the generation probability is larger than the rate at which outstanding requests can be satisfied. In such a case, requests are frequently ignored due to the chosen generation algorithm; in particular, note that in the case when a single hot spot is assigned to each PE, a mean request rate greater than the reciprocal of the number of stages in the path traversed by each packet ($2K$) is useless since requests can never be satisfied 100% of the time at such a rate.

3.3 Computing Averages

In computing the average queue lengths and wait buffer pool sizes output by the simulator, only the *requested* number of simulation steps was divided into the cumulative sums; i.e., the additional number of steps required to flush the system of outstanding requests was *not* accounted for in the averages. The rationale for this is based on the observation that each component of the simulated multicomputer is idle for at least $2K$ steps during a run due to the need to fill up the network with requests initially and the need to empty the network of outstanding requests prior to termination; the averages are computed to ignore these "pipeline"-oriented idle phases of the system. Although most simulation runs were performed for 1000 steps, the finalization time endured was in some cases as much as ten percent of the number of requested simulation steps.

3.4 Maxima vs. Means

Maximum queue lengths and wait buffer lengths represent the maximum length observed in a queue *at any time*; i.e., the values of the maxima at stage i are calculated *after* requests have been routed into stage i from the previous stage, but *before* requests have been routed out of stage i into the following stage. On the other hand, mean queue lengths and wait buffer lengths represent the mean length observed in the queues at each stage *at the end of a simulation step*; i.e, the cumulative sums used in computing means are updated at the end of a step *after* all routing has been performed. The rationale behind this slight difference is that the means are meant to convey the average queue lengths over all simulation steps, whereas transient maxima tend to effect the stage-level routing. The latter becomes especially apparent when the simulated system is constrained to have small, finite queues and wait buffers, as was explained in Section 3.1.

4 Results

This section presents an analysis of the output of over 100 runs of the simulator, grouped into four experiments:

1. General simulation of a 16-PE Ultracomputer.
2. General simulation of a 512-PE Ultracomputer.
3. Determination of critical queue sizes in a 512-PE Ultracomputer using the no-wait policy.
4. Test of the effect of hot spot spacing in a 512-PE Ultracomputer using the no-wait policy.

All runs were performed for 1000 steps, and each configuration that was to be simulated was simulated twice, using two different pseudo-random number generator seeds, to account for any possible instability in the generator. In addition, each run of the first two experiments was duplicated for the two request generation policies (wait vs. no-wait).

The simulation of the 16-PE system was performed mainly for testing, debugging and observation of the simulator code; therefore, attention will be focused in this section primarily on the simulation of the 512-PE systems. Since simulation of a 512-PE Ultracomputer for 1000 steps usually required 45 real minutes of an otherwise idle computer, results were generated in overnight batch jobs of groups of four to six runs.

4.1 General Simulation of a 16-PE System

For the general simulation of the 16-PE Ultracomputer, an initial configuration representing a "typical" or "normal" system was simulated, and then the characteristics (i.e, simulator input parameters) of this base configuration were perturbed for further simulation. This methodology allowed the verification of several broad and intuitive observations (discussed below) describing system performance in terms of system configuration or characteristics. Seven different systems were simulated in this manner:

1. A "normal" system.
2. A system with small, finite queues.
3. A system with a low probability of request generation (a light load).
4. A system with a high probability of request generation (a heavy load).
5. A system with very few hot spots.
6. A system with a large number of hot spots.
7. A system with a different hot spot-to-PE assignment scheme.

The parameter values used to configure these systems are presented in Table I. As was mentioned previously, each of these systems was simulated for both request generation policies and for two random number seeds, thus giving a total of 28 simulation runs. In addition, hot spot addresses were chosen randomly in all of these runs. Tables II and III present the major statistics resulting from these runs; Table II gives the statistics for the runs using the wait policy, while Table III gives the statistics for the no-wait policy runs. Each entry in these tables contains the two values of the particular statistic from the pair of corresponding runs on

Configuration	Total Hot Spots	Hot Spots per PE	Mean Max. Queue Length	Request Rate
Normal	2	0.95	∞	0.15
Small Queues	2	0.95	2	0.15
Light Load	2	0.95	∞	0.05
Heavy Load	2	0.95	∞	0.25
Few Hot Spots	1	0.95	∞	0.15
Many Hot Spots	4	1.90	∞	0.15
Modified Hot Spot Assignment	2	1.50	∞	0.15

Table I: Characteristics of Simulated 16-PE Systems.

Config.	Mean Requests per PE	Mean Steps per Req.	Max. Steps per Req.	Mean Queue Length	Max. Queue Length	Mean Wait Buffers	Max. Wait Buffers
Normal	56.563	10.046	12	0.187	3	0.057	3
	60.688	10.019	11	0.179	2	0.059	2
Small Queues	56.563	10.048	12	0.187	2	0.057	2
	60.688	10.019	11	0.179	2	0.059	2
Light Load	30.063	10.027	11	0.102	3	0.026	2
	32.688	10.004	11	0.102	2	0.020	2
Heavy Load	67.500	10.049	12	0.216	3	0.085	2
	71.563	10.027	11	0.208	2	0.078	2
Few Hot Spots	51.188	10.000	10	0.247	2	0.118	2
	64.313	10.000	10	0.293	2	0.168	3
Many Hot Spots	85.063	10.065	12	0.167	3	0.030	2
	86.063	10.025	11	0.142	3	0.031	2
Modified Assignment	76.750	10.081	12	0.241	3	0.089	2
	72.938	10.027	11	0.206	2	0.069	3

Table II: Performance of 16-PE Systems—Wait Policy.

Config.	Mean Requests per PE	Mean Steps per Req.	Max. Steps per Req.	Mean Queue Length	Max. Queue Length	Mean Wait Buffers	Max. Wait Buffers
Normal	130.313	10.169	12	0.369	4	0.255	4
	144.313	10.057	11	0.362	3	0.258	4
Small	129.313	10.274	16	0.376	2	0.241	2
Queues	144.313	10.092	13	0.366	2	0.245	2
Light	43.188	10.036	11	0.144	3	0.046	2
Load	47.375	10.013	11	0.144	3	0.034	2
Heavy	216.500	10.294	13	0.531	4	0.602	7
Load	226.875	10.117	11	0.498	3	0.545	5
Few Hot	119.250	10.000	10	0.444	2	0.480	4
Spots	152.500	10.000	10	0.461	2	0.668	5
Many Hot	151.813	10.143	12	0.280	4	0.095	3
Spots	146.250	10.065	12	0.228	3	0.081	3
Modified	140.875	10.160	12	0.394	3	0.257	3
Assignment	141.750	10.057	11	0.346	3	0.226	4

Table III: Performance of 16-PE Systems—No-Wait Policy.

two different seed values. The (global) queue and wait buffer statistics have been computed from the individual network stage statistics output by the simulator. The statistics were calculated only over the set of queues and buffers actually used for packet routing; in addition, the queue statistics incorporate both the PE-to-MM and the MM-to-PE stage statistics, since they were practically identical at each stage.

Table II demonstrates that in the presence of a wait policy, modifying other system parameters such as maximum queue length, request rate or number of hot spots produces no discernible change in system behavior. Queues of length three were adequate for all configurations, and no more than two extra steps of waiting were required during packet routing. On the other hand, Table III demonstrates the sensitivity of system performance to the chosen configuration in the presence of a no-wait policy. As expected, the amount of queue spaced used increases with increasing load, especially in the wait buffers. With small, finite-length queues, the number of routing steps required for some packets is as much as 60 percent above the minimum required. In addition, Table III confirms the theoretical relationship between request rate and total number of requests using a no-wait policy.

Configuration	Total Hot Spots	Hot Spots per PE	Mean Max. Queue Length	Request Rate
Normal	8	1.9	∞	0.15
Small Queues	8	1.9	3	0.15
Light Load	8	1.9	∞	0.05
Heavy Load	8	1.9	∞	0.25
Few Hot Spots	2	1.9	∞	0.15
Many Hot Spots	16	7.6	∞	0.15
Modified Hot Spot Assignment	8	4.5	∞	0.15

Table IV: Characteristics of Simulated 512-PE Systems.

Both tables demonstrate the ideal routing behavior which occurs in the presence of a single hot spot. There is never contention for queue space as all packets are pairwise combinable. Consequently, each packet may be routed in the minimum possible number of routing steps. The fact that queues of length two are used everywhere in this case is simply a side effect of the way statistics are gathered during the "forward" simulation of the system; in a purely synchronous system, single-element queues would suffice for routing to a single hot spot.

4.2 General Simulation of a 512-PE System

The general simulation of the 512-PE Ultracomputer was performed using the methodology of the previous section. Table IV presents the parameters used to configure the seven systems, while Table V (wait policy) and Table VI (no-wait policy) present the resulting statistics. Tables V and VI verify even more dramatically the observations made in the previous section regarding sensitivity of performance to configuration. The most striking phenomenon is the huge increase in both the mean and maximum number of routing steps required for packets using small, finite-length queues and a no-wait policy. The mean is almost double the minimum while some packets suffered as much as 900% degradation in routing delay. Both tables demonstrate the greater requirement for wait buffer space than for queue space. This seems reasonable if the opportunity for combining packets is great, since queue space is being traded for wait buffer space. Again, the ideal routing behavior in the presence of a single hot spot may be observed in the statistics for the first seed of the "Few Hot Spots" configuration; although two hot spots

Config.	Mean Requests per PE	Mean Steps per Req.	Max. Steps per Req.	Mean Queue Length	Max. Queue Length	Mean Wait Buffers	Max. Wait Buffers
Normal	59.484	20.028	22	0.338	3	0.495	6
	59.727	20.062	23	0.344	4	0.499	6
Small	59.205	20.170	25	0.345	3	0.497	3
Queues	59.406	20.247	25	0.353	3	0.503	3
Light	33.018	20.013	22	0.249	3	0.235	5
Load	33.170	20.030	22	0.253	3	0.237	5
Heavy	70.801	20.031	23	0.365	3	0.603	6
Load	71.092	20.075	25	0.372	4	0.606	6
Few Hot	59.625	20.000	20	0.526	2	1.733	8
Spots	59.428	20.048	22	0.542	3	1.734	9
Many Hot	109.527	20.252	27	0.352	5	0.457	8
Spots	108.697	20.259	26	0.352	4	0.453	8
Modified	90.961	20.043	24	0.403	4	0.775	7
Assignment	89.859	20.098	25	0.408	4	0.766	6

Table V: Performance of 512-PE Systems—Wait Policy.

were present, it appears that the chosen assignment of hot spots to PEs produced to completely disjoint trees of utilized routing hardware.

Figures 1–3 plot average queue lengths at each stage using the no-wait policy and reveal stage-dependent behavior of each system; Figure 1 shows mean PE-to-MM queue lengths, Figure 2 shows mean MM-to-PE queue lengths, and Figure 3 shows mean wait buffer sizes. In all of the plots, stage 1 represents the set of PEs, stages 0 through 8 represent the Omega network stages and stage 9 represents the set of MMs. Since the purpose of these plots is merely to illustrate global behavior patterns, the data have been taken from the set of runs for only one of the seed values; these data correspond to the *first* entries of statistics in Tables V and VI.

Figures 1, 2 and 3 demonstrate the increasing use of queue space and wait buffer space in the stages close to the memory modules, due to the convergence of packets on fewer switching ports. In the presence of small queues, it appears that the greatest proportion of routing delay is incurred by packets in their final approach to the memory modules. Figure 3 reveals the increased dependence on wait buffers for combining as the number of hot spots is decreased.

Config.	Mean Requests per PE	Mean Steps per Req.	Max. Steps per Req.	Mean Queue Length	Max. Queue Length	Mean Wait Buffers	Max. Wait Buffers
Normal	149.990	20.085	24	0.490	4	1.247	8
	148.916	20.236	26	0.500	5	1.244	9
Small	148.752	37.834	204	0.778	3	1.581	3
Queues	147.475	36.835	148	0.741	3	1.406	3
Light	49.930	20.022	22	0.312	3	0.403	6
Load	49.979	20.049	23	0.316	3	0.404	5
Heavy	247.342	20.182	25	0.582	4	1.856	9
Load	247.805	20.548	27	0.599	5	1.849	12
Few Hot	150.504	20.000	20	0.658	2	3.093	11
Spots	149.295	20.170	24	0.697	4	3.097	11
Many Hot	151.070	20.434	29	0.413	6	0.659	9
Spots	150.436	20.457	29	0.407	6	0.654	9
Modified	150.438	20.086	24	0.485	4	1.221	8
Assignment	149.830	20.216	25	0.496	4	1.216	8

Table VI: Performance of 512-PE Systems—No-Wait Policy.

4.3 Critical Queue Sizes in a 512-PE System

One important choice facing the designer of an Ultracomputer-like machine is the choice of queue sizes for the switchboxes. This section describes an experiment in which the critical queue sizes for three different request rates were sought. We define the critical queue size to be the largest queue size which produces the first discernible degradation of performance as the queue size is decreased from infinity. The simulations were performed on the "normal" configuration for 512 PEs using a no-wait policy. For each request rate simulated (0.05, 0.10, 0.15), four queue lengths around the estimated critical length were chosen for simulation, and once the critical length was verified, simulations of the resulting configuration on three more random seeds were run for good measure. All of the plots described below represent the results of the initial four runs; the extra runs on different seeds added nothing to the results.

Figure 4 plots mean PE-to-MM queue lengths, Figure 5 plots mean MM-to-PE queue lengths and Figure 6 plots mean wait buffer lengths, all for a request rate of 0.05. Figures 7–9 are the corresponding plots for a request rate of 0.10, while Figures 10-12 are the plots for a request rate of 0.15. The critical queue length for a request rate of 0.05 seems best derived from Figure 4, as Figures 5 and 6 reveal no significant performance variation. For queue lengths of three, four and five, performance is virtually identical; for queue lengths of two, queue utilization increases significantly for stages close to the memory modules, making two the critical queue size by our criteria. Similarly, Figure 7 together with Figure 9 suggests a critical queue length of four for a request rate of 0.10, while Figure 10 gives a critical queue length of five for a request rate of 0.15.

4.4 Effect of Hot Spot Spacing in a 512-PE System

In this section we describe an experiment in which the spacing of a fixed number of hot spots was varied across the set of memory modules in an attempt to test the sensitivity of performance to hot spot location. The simulations were performed on a "normal" configuration using a no-wait policy and a request rate of 0.15. Both infinite queues and finite queues of length five (the critical length for this request rate) were used, and the run for each configuration was performed twice using two different random seeds. Three different spacings of eight hot spots were simulated using the spacing-with-variance algorithm described in Section 2.1: A spacing of 3 ± 1 represents a "close" spacing, a spacing of 16 ± 2 represents a "medium" spacing while a spacing of 61 ± 2 represents a "wide" spacing in which the hot spots are distributed almost uniformly across the set of memory modules.

Figures 13–15 are the set of plots resulting from this experiment; only the statistics for the runs on the first seed are represented. Except for isolated points of divergence this experiment was unable to reveal any significant effect of the spacing of hot spots on system performance, except possibly that a "medium" spacing maximizes queue utilization in the center of the Omega network.

5 Conclusions

We have described the design of a simulator of an Ultracomputer in the presence of several hot spots using packet combination and various packet generation policies. The performance of the simulated system was analyzed based on empirical results obtained from several simulation experiments. We demonstrated that different system loads require different system configurations in order to obtain a near-optimal balance between cost and delay. However, our implementation was not able to reveal the effects, if any, of varying the relative placement of a fixed number of hot spots among the memory modules, for a fixed offered load.

As Kruskal and Snir observed [KS83], a purely mathematical performance analysis is intractable for all but the most simplistic models. A more realistic mathematical model which may allow derivation of improved general asymptotic results is the modeling of the Ultracomputer as a network of finite automata. By representing each switching element, PE and memory module as a finite automaton executing an appropriate stochastic process, the techniques of queueing theory applicable to networks of queues could be invoked for general and steady-state performance analysis. However, an analysis of large multiprocessors of around 512 PEs using such techniques will involve the solution of huge systems of descriptive equations.

References

[DoD83] *The Ada Programming Language Reference Manual.* US Department of Defense, US Government Printing Office, February 1983. ANSI/MIL-STD-1815A-1983.

[GGK83] A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph and M. Snir. The NYU Ultracomputer—designing an MIMD shared memory parallel computer. *IEEE Transactions on Computers*, 32(2):175–189, February 1983.

[KS83] C. P. Kruskal and M. Snir. The performance of multistage interconnection networks for multiprocessors. *IEEE Transactions on Computers*, 32(12):1091–1098, December 1983.

[PN85] G. F. Pfister and V. A. Norton. 'Hot spot' contention and combining in multistage interconnection networks. *IEEE Transactions on Computers*, 34(10):943–948, October 1985.

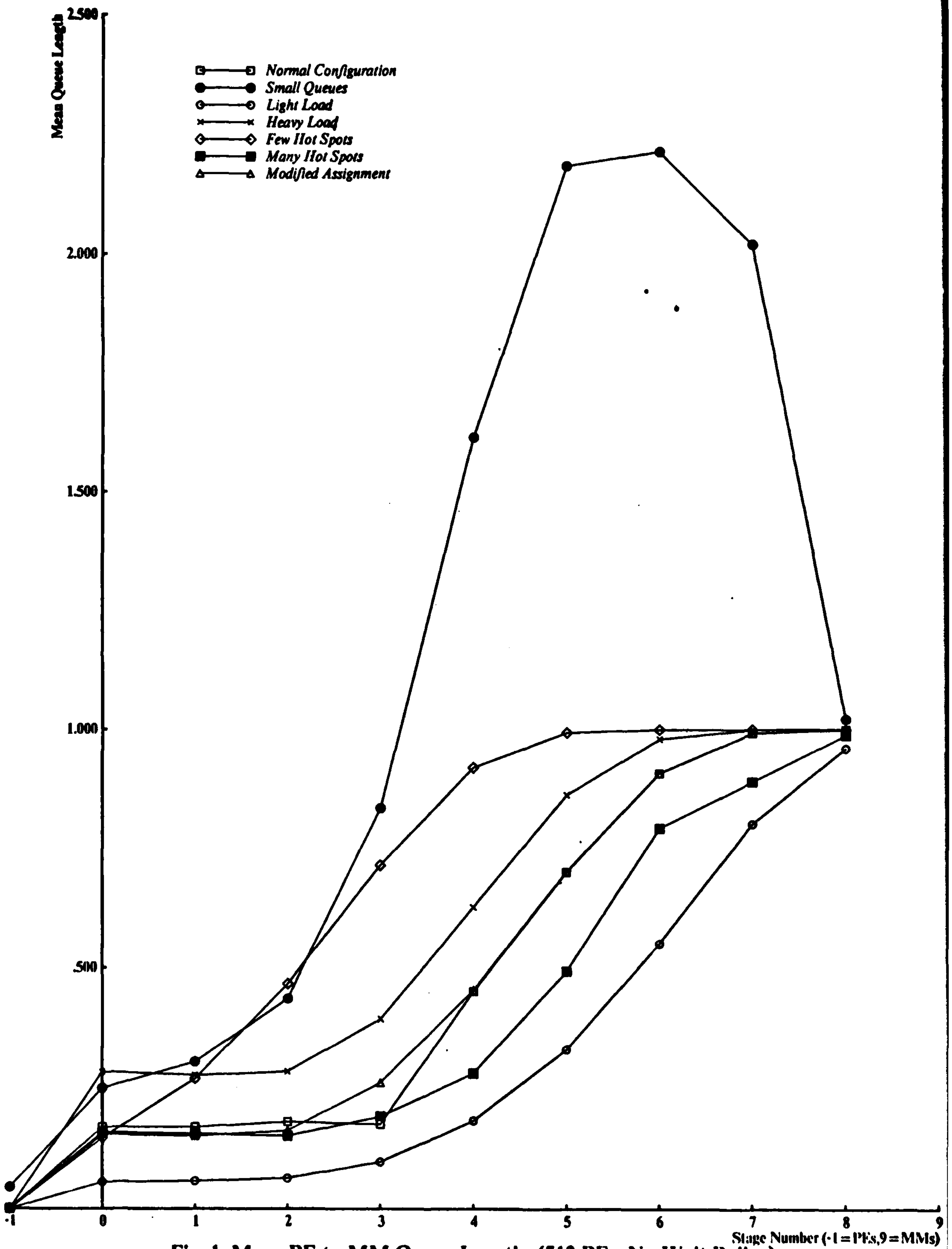


Fig. 1. Mean PE-to-MM Queue Lengths (512 PEs, No-Wait Policy)

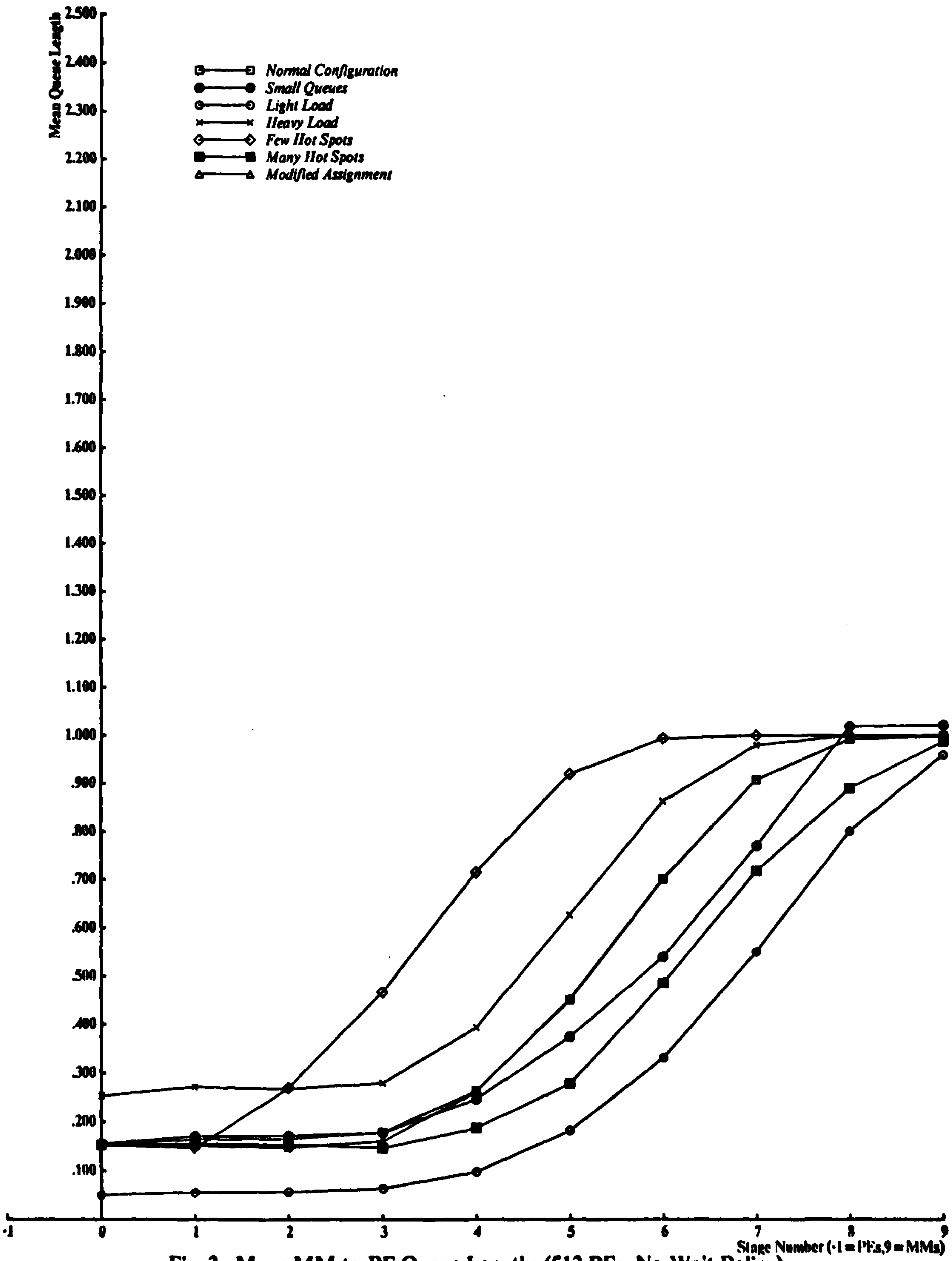


Fig. 2. Mean MM-to-PE Queue Lengths (512 PEs, No-Wait Policy)

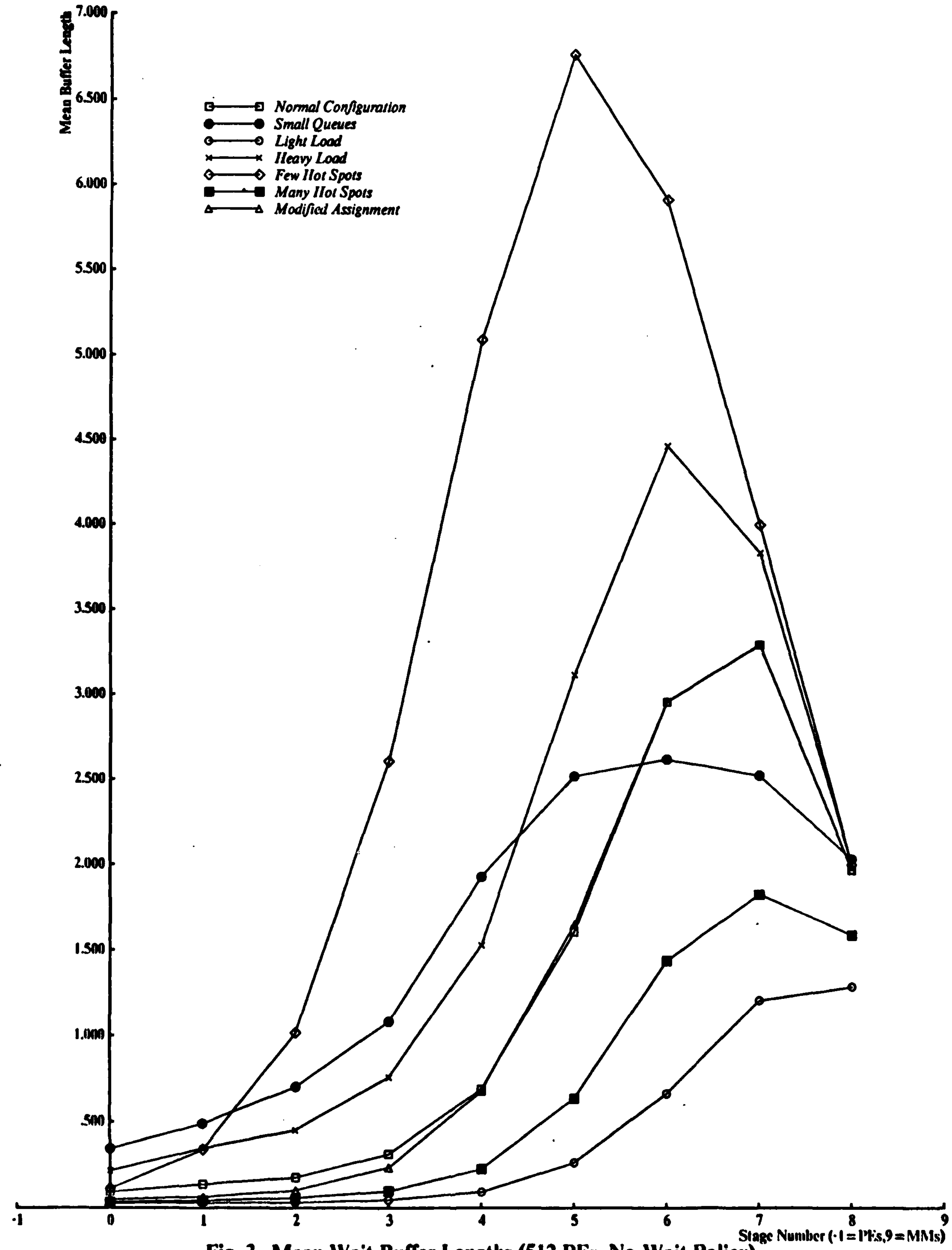


Fig. 3. Mean Wait Buffer Lengths (512 PEs, No-Wait Policy)

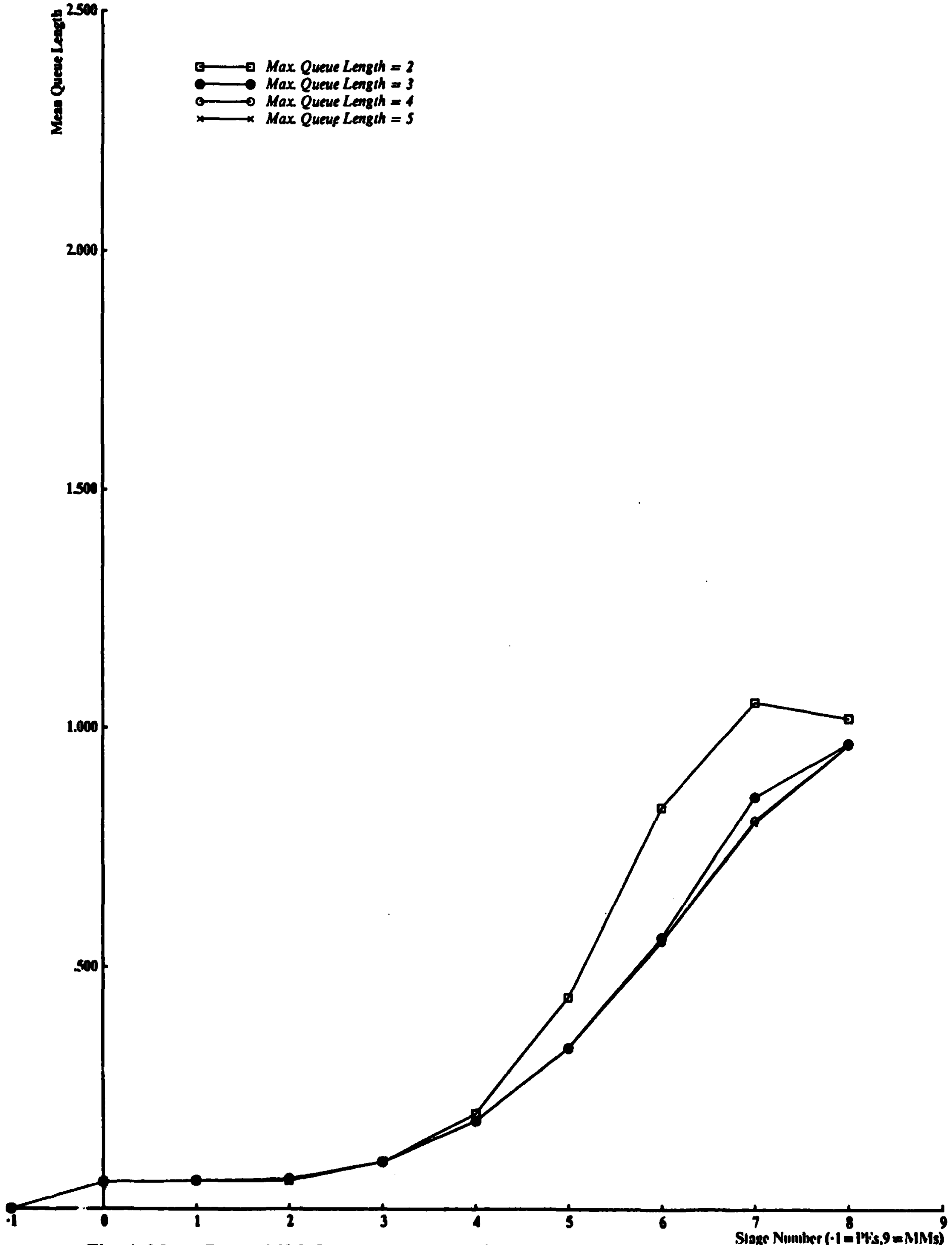


Fig. 4. Mean PE-to-MM Queue Lengths (512 PEs, No-Wait Policy, Request Rate 0.05)

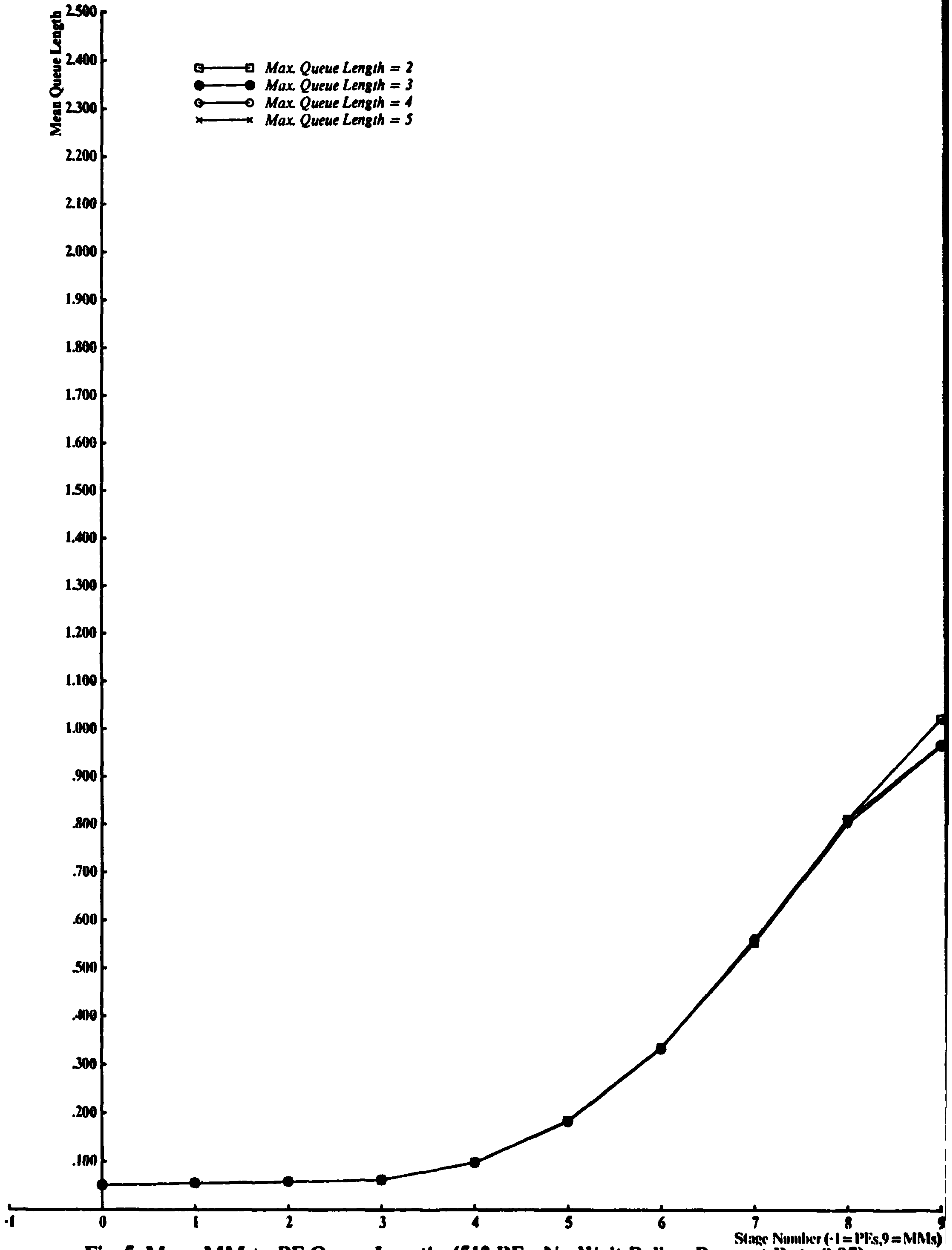


Fig. 5. Mean MM-to-PE Queue Lengths (512 PEs, No-Wait Policy, Request Rate 0.05)

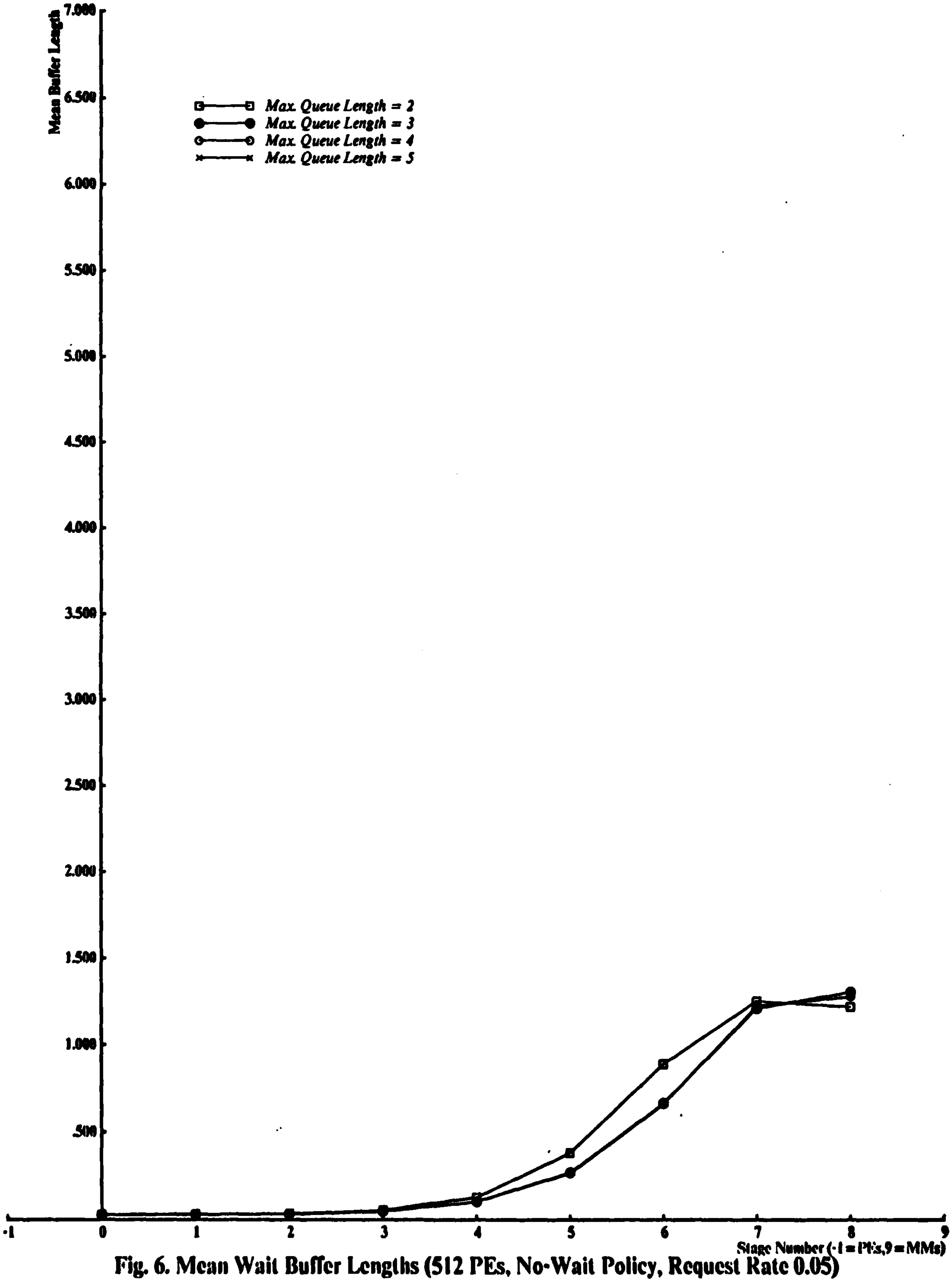


Fig. 6. Mean Wait Buffer Lengths (512 PEs, No-Wait Policy, Request Rate 0.05)

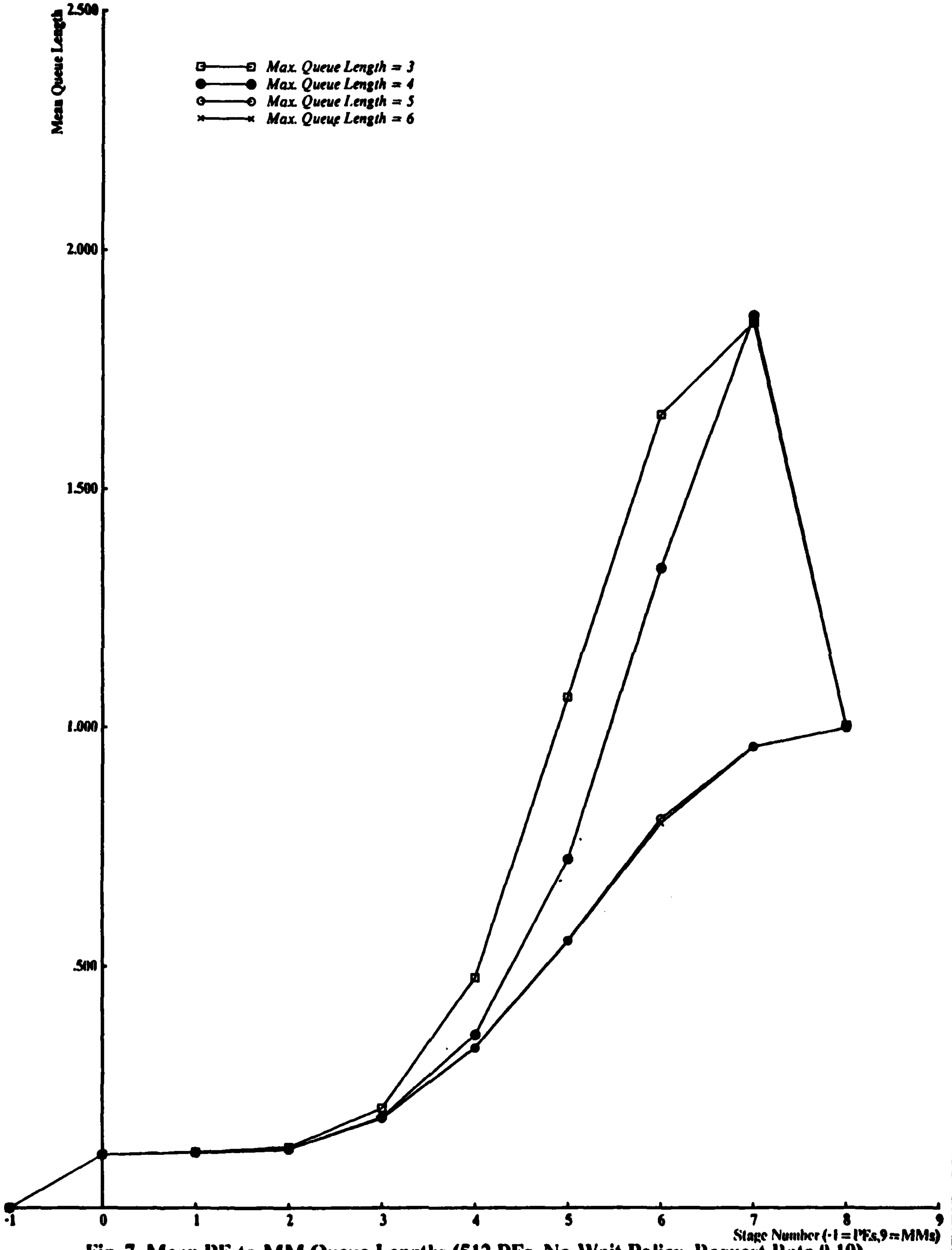


Fig. 7. Mean PE-to-MM Queue Lengths (512 PEs, No-Wait Policy, Request Rate 0.10)

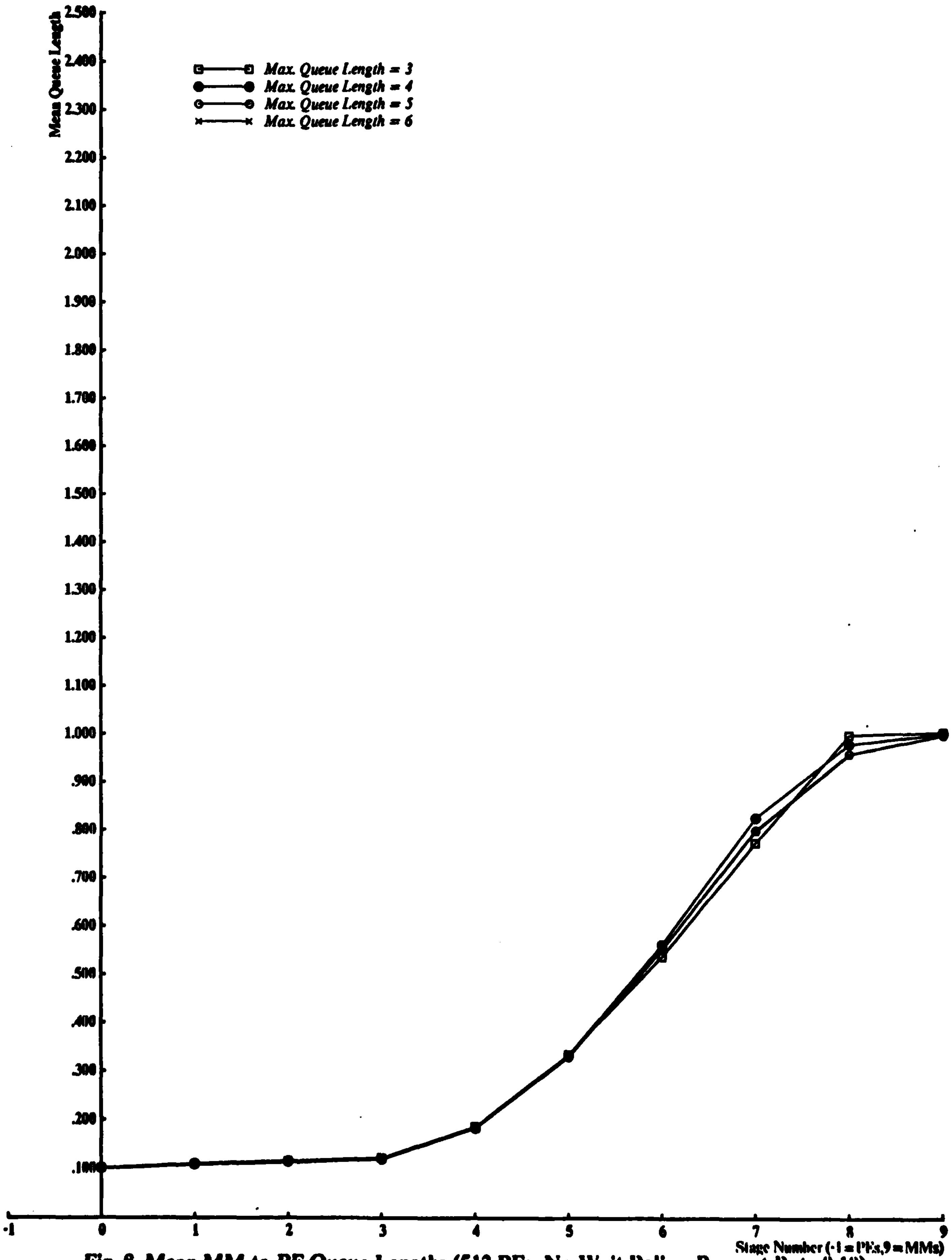


Fig. 8. Mean MM-to-PE Queue Lengths (512 PEs, No-Wait Policy, Request Rate 0.10)

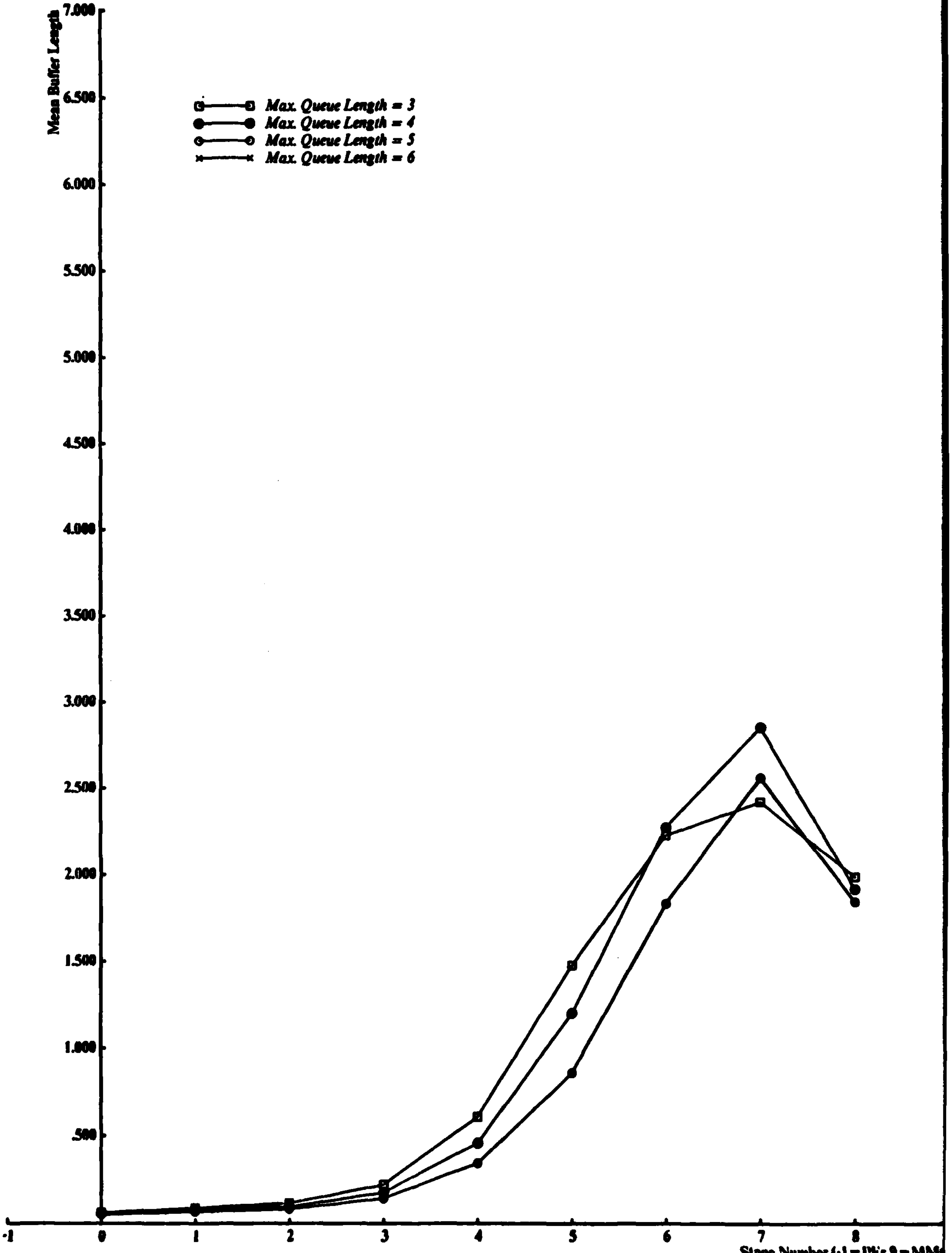


Fig. 9. Mean Wait Buffer Lengths (512 PEs, No-Wait Policy, Request Rate 0.10)

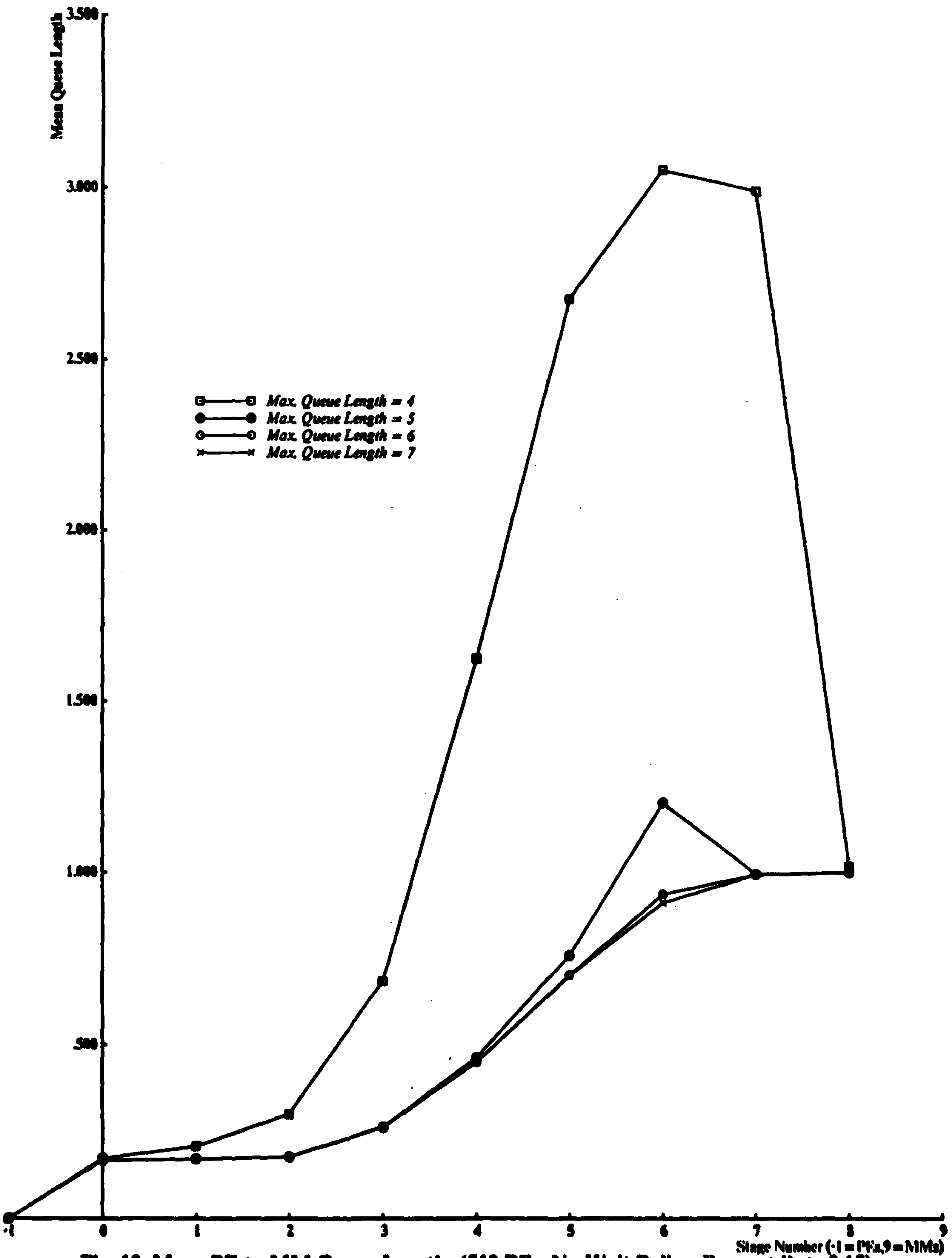


Fig. 10. Mean PE-to-MM Queue Lengths (512 PEs, No-Wait Policy, Request Rate 0.15)

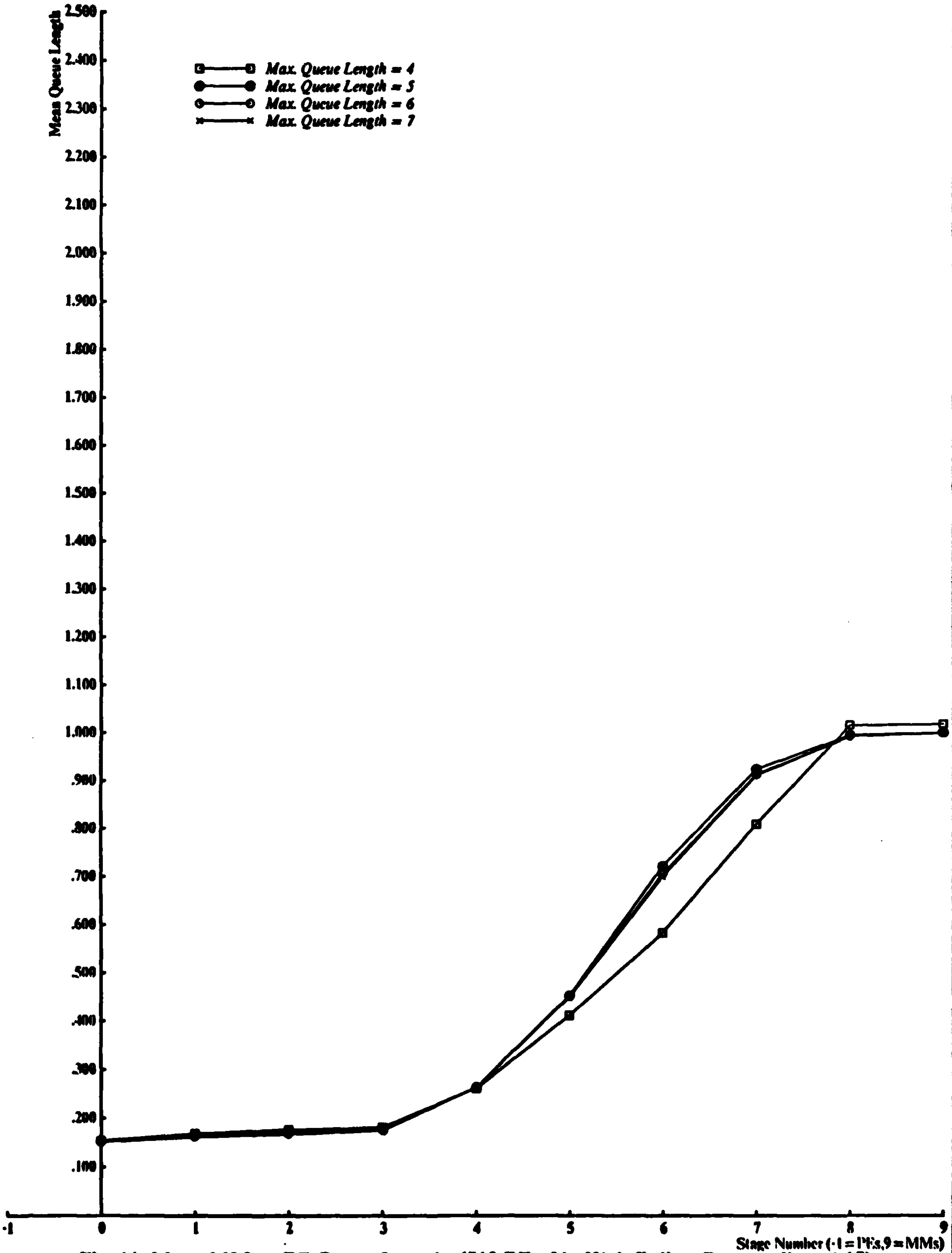


Fig. 11. Mean MM-to-PE Queue Lengths (512 PEs, No-Wait Policy, Request Rate 0.15)

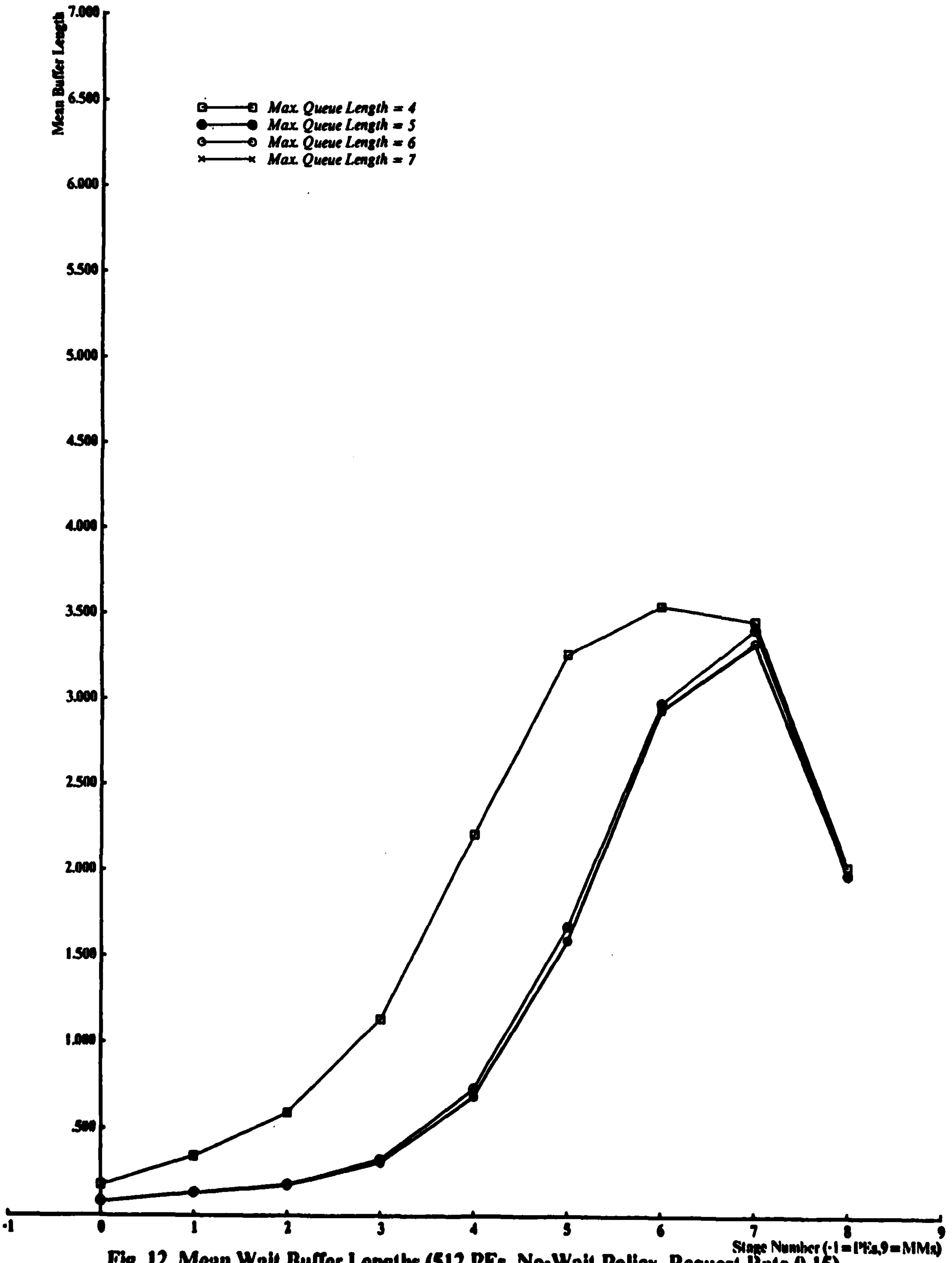


Fig. 12. Mean Wait Buffer Lengths (512 PEs, No-Wait Policy, Request Rate 0.15)

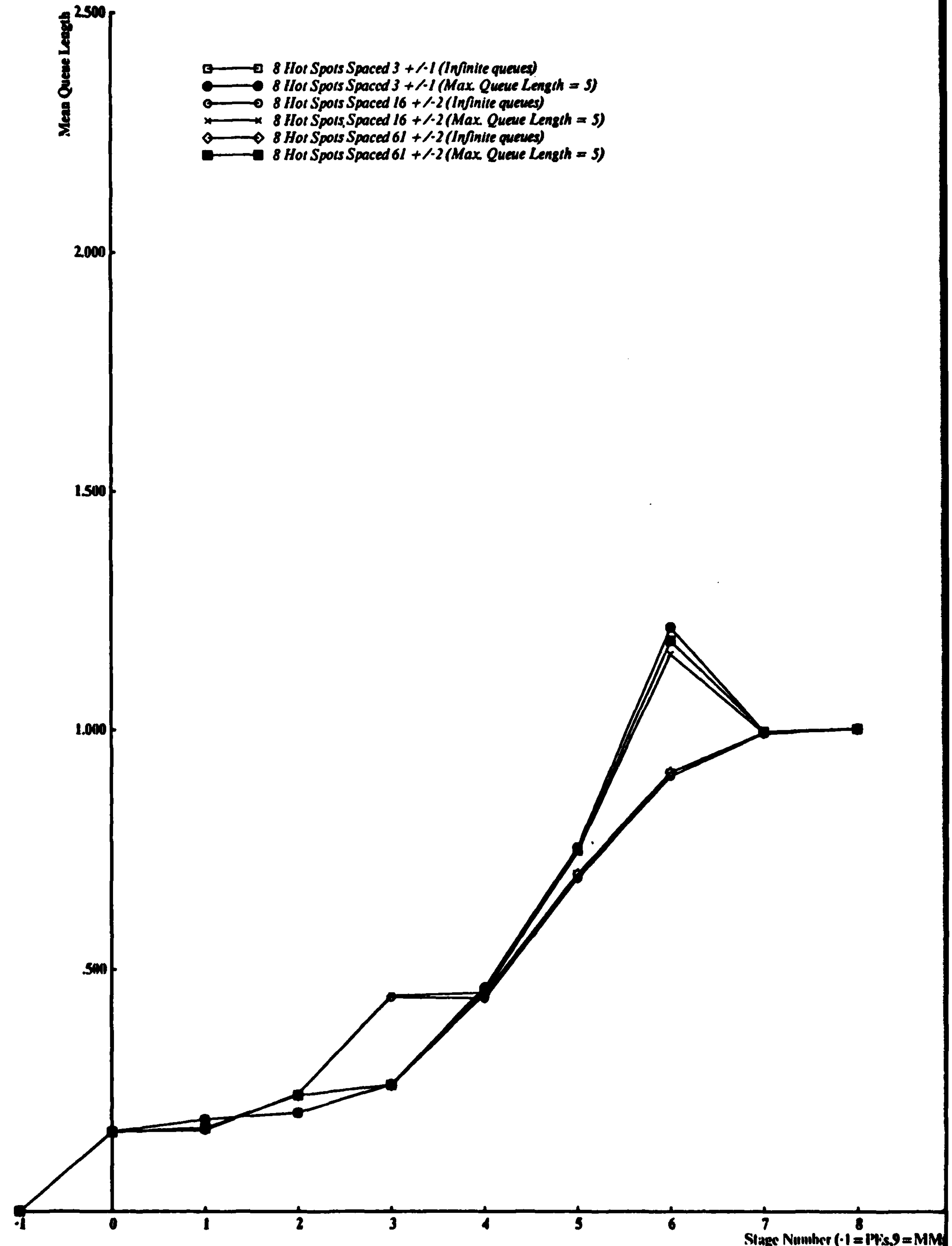


Fig. 13. Mean PE-to-MM Queue Lengths (512 PEs, No-Wait Policy, Request Rate 0.15)

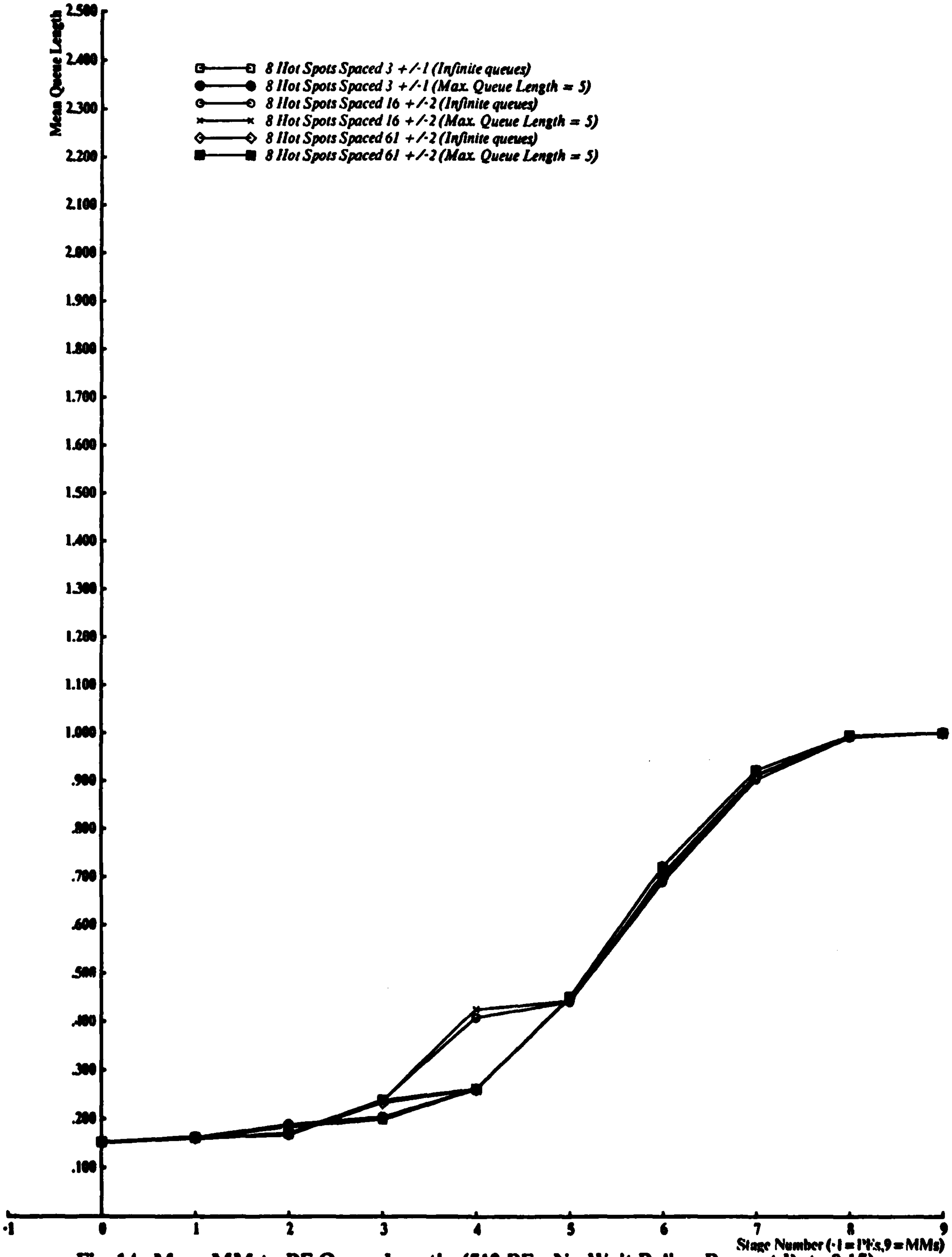


Fig. 14. Mean MM-to-PE Queue Lengths (512 PEs, No-Wait Policy, Request Rate 0.15)

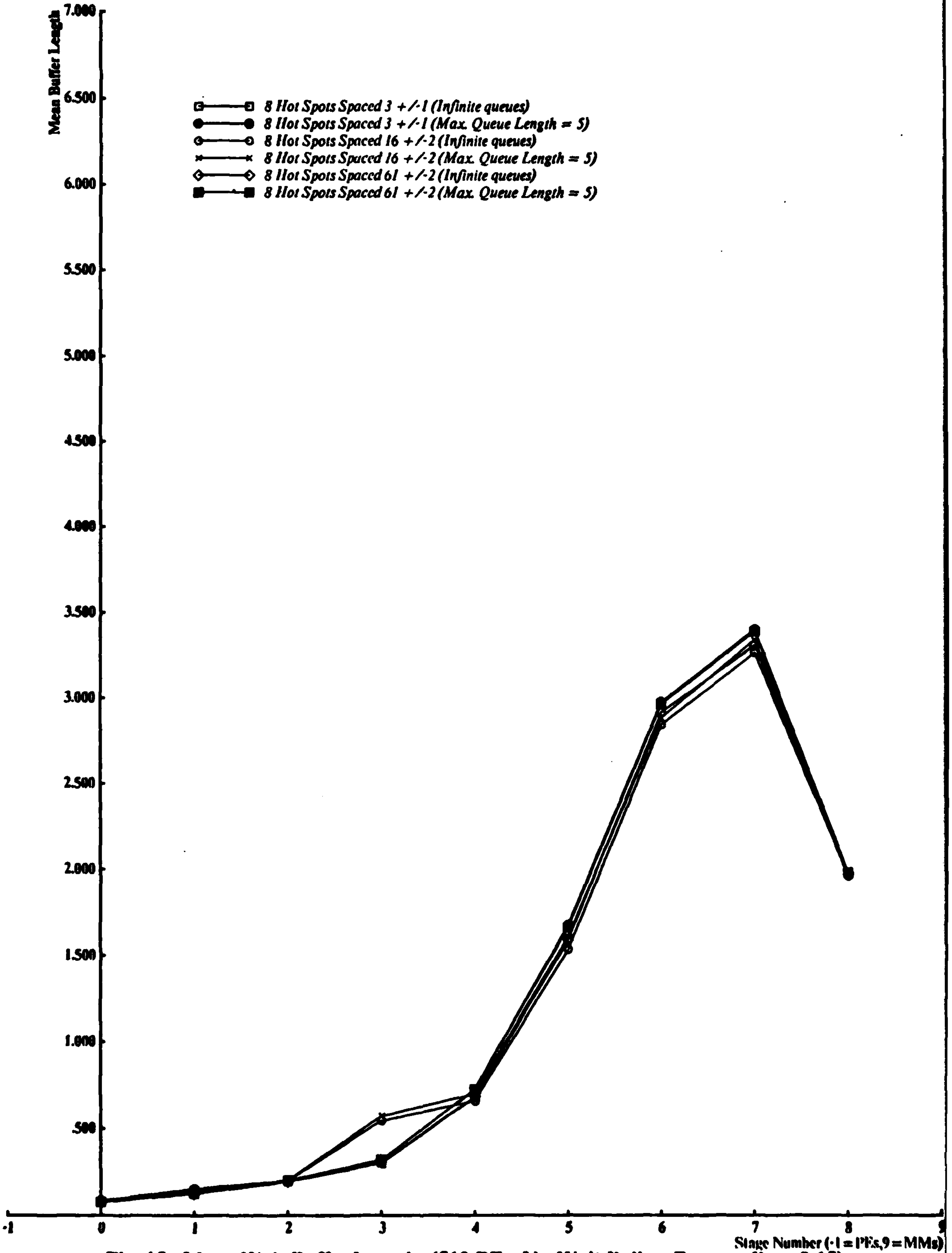


Fig. 15. Mean Wait Buffer Lengths (512 PEs, No-Wait Policy, Request Rate 0.15)

END

11-86

DTIC